PanOS Bootstrapper Utility Documentation

Release 0.4

Palo Alto Networks

Jun 05, 2020

CONTENTS:

1 Quickstart **1**
- 1.1 Docker . . . 1
- 1.2 Standalone . . . 1

2 Bootstrap on Amazon Web Services **3**
- 2.1 Creating an AWS Access Key . . . 3
- 2.2 Next Steps . . . 3

3 Bootstrap on Microsoft Azure **5**
- 3.1 Finding your Access Key . . . 5
- 3.2 Next Steps . . . 5

4 Bootstrap on Google Cloud Platform **7**
- 4.1 Finding your Project Id . . . 7
- 4.2 Granting an OAuth Token API Access . . . 7
- 4.3 Next Steps . . . 8

5 Bootstrap on Linux KVM **9**
- 5.1 More Information . . . 9

6 Bootstrap on Openstack **11**
- 6.1 Example Instance Creation . . . 11
- 6.2 Bootstrapping with the Openstack Horizon UI . . . 11
- 6.3 More Information . . . 12

7 Bootstrap on VMWare ESXi **13**
- 7.1 More Information . . . 13

8 Using the API **15**
- 8.1 Generate a minimal Bootstrap Archive . . . 15
- 8.2 Controlling the output format . . . 15
- 8.3 Using JSON Input . . . 15
- 8.4 List available templates . . . 16
- 8.5 List Init-Cfg Templates . . . 16
- 8.6 Show the contents of a template . . . 17
- 8.7 To show required variables in a set of templates . . . 17
- 8.8 Building a bootstrap package . . . 18
- 8.9 Building a Bootstrap Package with a custom bootstrap.xml . . . 18

9 Using the bootstrapper-cli **21**

10 Testing **23**
10.1 Executing Tests 23
10.2 Example test output 23

11 About **25**

12 Architecture **27**

13 Contributing **29**

14 Indices and tables **31**

CHAPTER

ONE

QUICKSTART

1.1 Docker

The fastest way to start this tool is using Docker. New container images are built periodically and will always be up to date.

```
docker build -t panos_bootstrapper:v0.4 .
docker run -p 5002:5000 -e PYTHONUNBUFFERED=0 nembery/panos_bootstrapper
```

1.2 Standalone

For local development, start the tool directly using these commands:

```
export FLASK_APP=./bootstrapper/bootstrapper.py
flask run --host=0.0.0.0 --port=5002
```

This will start the API and listen on all interfaces on port 5002. Browsing to http://localhost:5002 will show the OpenAPI 2.0 documentation.

CHAPTER

TWO

BOOTSTRAP ON AMAZON WEB SERVICES

Bootstrapper can build S3 buckets on Amazon using custom bootstrap.xml and init-cfg templates. In order to create files and folders, Bootstrapper needs your Access Key and Access Secret. This information is never stored on disk.

2.1 Creating an AWS Access Key

To create an access key for your AWS account root user

Use your AWS account email address and password to sign in to the AWS Management Console as the AWS account root user.

Note: If you previously signed in to the console with IAM user credentials, your browser might remember this preference and open your account-specific sign-in page. You cannot use the IAM user sign-in page to sign in with your AWS account root user credentials. If you see the IAM user sign-in page, choose Sign-in using root user credentials near the bottom of the page to return to the main sign-in page. From there, you can type your AWS account email address and password.

In the IAM navigation pane, choose Users.

Choose the name of the preferred user, and then choose the Security credentials tab.

If needed, expand the Access keys section.

Choose Create New Access Key. Then choose Download Key File to save the access key ID and secret access key to a file on your computer. After you close the dialog box, you can't retrieve this secret access key again.

Warning: A root access key grants full programmatic access to your resources, meaning that it should be guarded as carefully as the root sign-in credentials for your account.

2.2 Next Steps

Refer to the official documentation here: Bootstrap the VM-Series Firewall on AWS

Note: You may need to open a new browser window to follow links to external sites when viewing these docs in an embedded environment like Bootstrapper-UI

CHAPTER

THREE

BOOTSTRAP ON MICROSOFT AZURE

Bootstrapper can build File Shares on Azure using custom bootstrap.xml and init-cfg templates. In order to create files and folders, Bootstrapper needs your Storage Account Name and Storage Access Key. This information is never stored on disk.

3.1 Finding your Access Key

To find your access key for Azure:

Use your Azure account email address and password to sign in to the Azure Portal.

Choose a valid storage account from the Dashboard in the **resources** section

In the **settings** section, chose the **Access Keys** option.

Take note of the **Storage Account Name**

Copy either **key1** or **key2**.

> **Warning:** An access key grants full programmatic access to your resources, meaning that it should be guarded as carefully as the root sign-in credentials for your account.

3.2 Next Steps

Refer to the official documentation here: Bootstrap the VM-Series Firewall in Azure

---

Note: You may need to open a new browser window to follow links to external sites when viewing these docs in an embedded environment like Bootstrapper-UI

CHAPTER

FOUR

BOOTSTRAP ON GOOGLE CLOUD PLATFORM

Bootstrapper can build buckets on GCP using custom bootstrap.xml and init-cfg templates. In order to create files and folders, Bootstrapper needs your OAuth Token and Project Id. This information is never stored on disk.

4.1 Finding your Project Id

To find your project id and Oauth Token:

Use your GCP account email address and password to sign in to the GCP Cloud Portal.

From the Cloud Console Home screen, the **Project ID** can be found on the **Project Info** widget.

4.2 Granting an OAuth Token API Access

In order to access the GCP APIs, Bootstrapper needs a valid OAuth Token. To generate an OAuth token:

Browse to the OAuth Playground.

In Step 1: **Select & authorize APIs** scroll down to the **Cloud Storage JSON API v1**.

Click on the caret to open the **Cloud Storage JSON API v1** and highlight the **https://www.googleapis.com/auth/devstorage.read_write** option.

Click the **Authorize API** button.

This will open a **Choose Account** page. Select an appropriate Google Account.

Click the **Allow** button to grant access to the OAuth Playground.

You will be redirected back to the OAuth Playground with **Step 2** opened. Click the **Exchange authorization code for tokens** button.

Copy the value of the **Access Token** field. This token will only be valid for 1 hour.

Note: You may need to click on **Step 2** again to find the **Access Token** field. The OAuth playground will occassionally hide this option and open **Step 3**. **Step 3** is not needed for this application.

4.3 Next Steps

Refer to the official documentation for bootstrapping a VM-Series firewall here: Bootstrapping Workflow

Note: You may need to open a new browser window to follow links to external sites when viewing these docs in an embedded environment like Bootstrapper-UI

CHAPTER

FIVE

BOOTSTRAP ON LINUX KVM

Bootstrapper can build ISO images using custom bootstrap.xml and init-cfg templates. This ISO can then be attached to the 'cdrom' slot on a KVM virtual machine to begin the bootstrap process.

5.1 More Information

To complete the bootstrap process, refer to the official documentation:

Bootstrap the VM-Series Firewall on KVM

Note: You may need to open a new browser window to follow links to external sites when viewing these docs in an embedded environment like Bootstrapper-UI

CHAPTER

SIX

BOOTSTRAP ON OPENSTACK

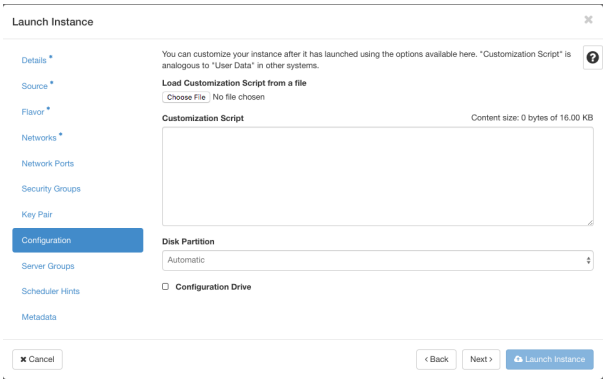
Bootstrapper can build tar.gz archives using custom bootstrap.xml and init-cfg templates. This archive can then be attached to the instance using the user-data flag. Most newer versions of Openstack require base64 encoded user-data archives. Panos-bootstrapper supports both tar.gz formatted archives as well as base64 encoded tar.gz archives. The Openstack deployment option uses the encoded tar.gz format by default.

6.1 Example Instance Creation

```
curl -J -O  -X POST -d "hostname=panos-vm-01" -d "archive_type=encoded_tgz" ␣
↪localhost:5001/generate_bootstrap_package

nova boot --config-drive true --image <pan-os-image-file-name> --flavor <flavor> --
↪user-data ./panos-vm-01.tgz.base64
--security-groups <security-group> --nic net-id=<mgmt nic net-id> --nic net-id=<eth1␣
↪nic net-id>
--nic net-id=<eth2 nic net-id> panos-vm-01
```

6.2 Bootstrapping with the Openstack Horizon UI



Bootstrapping a VM-Series NGFW using the Openstack Horizon can be done by first creating the archive using the 'encoded_tgz' deployment type option. Then choosing that file in the 'Load Customization Script from a file' dialog in the Instance creation pop-up. You must also ensure 'Configuration Drive' is selected.

6.3 More Information

To complete the bootstrap process, refer to the official documentation:

Bootstrap the VM-Series Firewall on KVM in OpenStack

Note: You may need to open a new browser window to follow links to external sites when viewing these docs in an embedded environment like Bootstrapper-UI

CHAPTER

SEVEN

BOOTSTRAP ON VMWARE ESXI

Bootstrapper can build ISO images using custom bootstrap.xml and init-cfg templates. This ISO can then be attached to the 'cdrom' slot on during virtual machine provisiojing to begin the bootstrap process.

7.1 More Information

To complete the bootstrap process, refer to the official documentation:

Bootstrap the VM-Series Firewall on ESXi with an ISO

Note: You may need to open a new browser window to follow links to external sites when viewing these docs in an embedded environment like Bootstrapper-UI

CHAPTER

EIGHT

USING THE API

API documentation is included in OpenAPI (currently swagger 2.0) format. A simple swagger API viewer is included in the root directory by browsing to http://bootstrapper_host:5000/, where bootstrapper_host is the host where the bootstrapper service is running.

Some examples are given below:

8.1 Generate a minimal Bootstrap Archive

```
local:~ operator$ curl -J -O  -X POST -d "hostname=PANOS-01"  localhost:5001/generate_
↪bootstrap_package
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                               Dload  Upload   Total   Spent    Left  Speed
100  138M  100  138M  100    22  17.9M      2  0:00:11  0:00:07  0:00:04 30.6M
curl: Saved to filename 'PANOS-TEST-01.zip'
```

8.2 Controlling the output format

```
local:~ operator$ curl -J -O  -X POST -d "hostname=PANOS-TEST-01" -d "archive_type=iso
↪"  localhost:5001/generate_bootstrap_package
   % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                  Dload  Upload   Total   Spent    Left  Speed
 100  138M  100  138M  100    39  37.5M     10  0:00:03  0:00:03 --:--:-- 37.5M
 curl: Saved to filename 'PANOS-TEST-01.iso'
```

8.3 Using JSON Input

```
local:~ operator$ curl -J -O  -X POST -d '{"hostname": "PANOS-TEST-02", "archive_type
↪": "iso"}' -H "Content-Type: application/json" localhost:5001/generate_bootstrap_
↪package
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                  Dload  Upload   Total   Spent    Left  Speed
100  138M  100  138M  100    52  32.9M     12  0:00:04  0:00:04 --:--:-- 34.6M
curl: Saved to filename 'PANOS-TEST-02.iso'
```

8.4 List available templates

```
local:~ operator$ curl  http://localhost:5000/list_templates | python -m json.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   458  100   458    0     0  39031      0 --:--:-- --:--:-- --:--:-- 41636
{
    "status_code": 200,
    "success": true,
    "templates": [
        {
            "description": "No Bootstrap.xml Required",
            "name": "None",
            "type": "bootstrap"
        },
        {
            "description": "Default Bootstrap template",
            "name": "Default Bootstrap.xml",
            "type": "bootstrap"
        },
        {
            "description": "Imported Template",
            "name": "GKE_Bootstrap",
            "type": "bootstrap"
        },
        {
            "description": "Imported Template",
            "name": "VMWare_Bootstrap",
            "type": "bootstrap"
        },
        {
            "description": "Imported Template",
            "name": "AWS_Bootstrap",
            "type": "bootstrap"
        }
    ]
}
```

8.5 List Init-Cfg Templates

```
local:~ operator$ curl  http://localhost:5000/list_init_cfg_templates | python -m␣
↪json.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   413  100   413    0     0  30134      0 --:--:-- --:--:-- --:--:-- 31769
{
    "status_code": 200,
    "success": true,
    "templates": [
        {
            "description": "PAN-OS Version 8.0 Init-Cfg",
            "name": "Default Init-Cfg",
            "type": "init-cfg"
        }
```

(continues on next page)

(continued from previous page)

```
    ]
}
```

8.6 Show the contents of a template

```
local:~ operator$ curl -X POST -d "template_name=Default Init-Cfg"  http://
↪localhost:5001/get_template
type={{ dhcp_or_static }}
ip-address={{ ip_address }}
default-gateway={{ default_gateway }}
netmask={{ netmask }}
ipv6-address={{ ipv6_address }}
ipv6-default-gateway={{ ipv6_default_gateway }}
hostname={{ hostname }}
panorama-server={{ panorama_server }}
panorama-server-2={{ panorama_server_2 }}
tplname={{ tpl_name }}
dgname={{ dg_name }}
dns-primary={{ dns_primary }}
dns-secondary={{ dns_secondary }}
op-command-modes={{ op_command_modes }}
dhcp-send-hostname={{ dhcp_send_hostname }}
dhcp-send-client-id={{ dhcp_send_client_id }}
dhcp-accept-server-hostname={{ dhcp_accept_server_hostname }}
dhcp-accept-server-domain={{ dhcp_accept_server_domain }}
vm-auth-key={{ vm_auth_key }}
```

This template only defines one variable. In this case *hostname* is declared as a variable. To use this template in a bootstrap package, you must supply a *hostname* variable to the *generate_bootstrap_package* API.

8.7 To show required variables in a set of templates

```
local:~ operator$ curl -X POST -d '{"init_cfg_template": "init-cfg-hostname"}' -H
↪"Content-Type: application/json" http://localhost:5000/get_bootstrap_variables |␣
↪python -m json.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   188  100   146  100    42  11718   3371 --:--:-- --:--:-- --:--:-- 12166
{
    "payload": {
        "archive_type": "iso",
        "deployment_type": "kvm",
        "hostname": "",
        "init_cfg_template": "init-cfg-hostname"
    },
    "status_code": 200,
    "success": true
}
```

This example uses the *get_bootstrap_variables* API to return the required payload for the desired templates. In this case, the keys listed in the payload dictionary will be required to build a bootstrap package using only the *init-cfg-hostname* template.

8.8 Building a bootstrap package

```
curl -X POST -d '{ "archive_type": "iso", "deployment_type": "kvm", "hostname": "NGFW-
↪001", "init_cfg_template": "init-cfg-hostname"}' -H "Content-Type: application/json
↪"  http://localhost:5000/generate_bootstrap_package -o NGFW.iso
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  380k  100  380k  100   116  10.6M   3319 --:--:-- --:--:-- --:--:-- 10.9M
local:~ operator$ hdiutil mount NGFW.iso
/dev/disk7                                                  /Volumes/bootstrap 5
local:~ operator$ cd /Volumes/bootstrap\ 5/
local:bootstrap 5 operator$ ls
config              content         license         software
local:bootstrap 5 operator$ cd config/
local:config operator$ ls
init-cfg.txt
local:config operator$ cat init-cfg.txt
type=dhcp
ip-address=
default-gateway=
netmask=
hostname=NGFW-001
dns-primary=
panorama-server=
dgname=
tplname=
vm-auth-key=
```

In this example, we took the output of the *get_bootstrap_variables* API call, entered our desired *hostname* (NGFW-001 in this case) and POSTed that information to the *generate_bootstrap_package* API. This returned an ISO image with the desired init-cfg template compiled with our variables. Attaching this ISO to a factory default PAN-OS firewall will result in the firewall booting up with the NGFW-001 hostname configured at boot.

8.9 Building a Bootstrap Package with a custom bootstrap.xml

In the previous example, we only built a package that included the init-cfg.txt file. However, you can also include a complete firewall configuration using a *bootstrap.xml* file.

Once again, let's get all required variables for our selected templates: *note that we've included a *bootstrap_template* parameters with the value of a bootstrap template name.

```
local:curl -X POST -d '{"init_cfg_template": "Default Init-Cfg", "bootstrap_template
↪": "Default Bootstrap.xml"}' -H "Content-Type: application/json"  http://
↪localhost:5000/get_bootstrap_variables | python -m json.tool
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   438  100   345  100    93  19049   5135 --:--:-- --:--:-- --:--:-- 19166
{
    "payload": {
        "archive_type": "iso",
        "bootstrap_template": "Default Bootstrap.xml",
        "default_next_hop": "",
        "deployment_type": "kvm",
        "ethernet1_1_profile": "",
        "ethernet2_1_profile": "",
```

(continues on next page)

(continued from previous page)

```
        "hostname": "",
        "init_cfg_template": "Default Init-Cfg",
        "management_gateway": "",
        "management_ip": "",
        "management_mask": "",
        "timezone": ""
    },
    "status_code": 200,
    "success": true
}
```

This output now includes the variables required for both the init-cfg template as well as the bootstrap template.

```
local:~ operator$ curl -X POST -d '{ "archive_type": "iso", "bootstrap_template":
↪"Default Bootstrap.xml", "default_next_hop": "10.0.1.1", "deployment_type": "kvm",
↪"ethernet1_1_profile": "PING", "ethernet2_1_profile": "PING", "hostname": "NGFW-003
↪", "init_cfg_template": "Default Init-Cfg", "management_gateway": "10.0.1.1",
↪"management_ip": "10.0.1.129", "management_mask": "255.255.255.0", "timezone":
↪"NewYork"}' -H "Content-Type: application/json"  http://localhost:5000/generate_
↪bootstrap_package -o NGFW-003.iso
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  394k  100  394k  100   385  7857k   7678 --:--:-- --:--:-- --:--:-- 7880k
local:~ operator$ hdiutil mount NGFW-003.iso
/dev/disk2                                              /Volumes/bootstrap
local:~ operator$ cat /Volumes/bootstrap/config/init-cfg.txt
type=dhcp
hostname=NGFW-003
dns-primary=
panorama-server=
dgname=
tplname=
vm-auth-key=
local:~ operator$ cat /Volumes/bootstrap/config/bootstrap.xml | grep hostname
          <hostname>NGFW-003</hostname>
```

CHAPTER

NINE

USING THE BOOTSTRAPPER-CLI

If you do not want to have the bootstrapper service always available via a REST interface, you can use the bootstrapper-cli interface.

```
cat /tmp/bootstrapper_cli_example.yaml
---
dhcp_or_static: dhcp-client
ip_address:
default_gateway:
netmask:
ipv6_address:
ipv6_default_gateway:
hostname: my-example-hostname
panorama_server:
panorama_server_2:
tpl_name:
dg_name:
dns_primary:
dns_secondary:
op_command_modes:
dhcp_send_hostname:
dhcp_send_client_id:
dhcp_accept_server_hostname:
dhcp_accept_server_domain:
vm_auth_key:
auth_code: VALID_AUTHCODE_HERE
```

and launch with:

```
docker run -it --rm -v "$(pwd):/var/tmp" -w /var/tmp nembery/panos_bootstrapper␣
↪bootstrap.sh build_bootstrap_iso bootstrapper_cli_example.yaml
```

You can also use this interface to build bootstrap archives in all the various public clouds. For AWS for example:

```
docker run -it --rm -v "$(pwd):/var/tmp" -w /var/tmp -e AWS_LOCATION=$(echo $AWS_
↪LOCATION) -e AWS_ACCESS_KEY=$(echo $AWS_ACCESS_KEY) -e AWS_SECRET_KEY=$(echo $AWS_
↪SECRET_KEY) nembery/panos_bootstrapper  bootstrap.sh build_bootstrap_aws␣
↪bootstrapper_cli_example.yaml
```

Azure is similar. Set the appropriate environment variables then run the build_bootstrap_azure command:

```
docker run -it --rm -v "$(pwd):/var/tmp" -w /var/tmp -e AZURE_STORAGE_ACCESS_KEY=
↪$(echo $AZURE_STORAGE_ACCESS_KEY) -e AZURE_STORAGE_ACCOUNT=$(echo $AZURE_STORAGE_
↪ACCOUNT) nembery/panos_bootstrapper  bootstrap.sh build_bootstrap_azure␣
↪bootstrapper_cli_example.yaml
```

CHAPTER

TEN

TESTING

Bootstrapper uses the pytest framework for unit testing

10.1 Executing Tests

execute all tests with:

```
cd bootstrapper
python -m pytest tests
```

You can also call it like this if desired:

```
PYTHONPATH=. pytest -v
```

10.2 Example test output

```
(panos-bootstrapper) DFWMACK0AJHTDG:panos-bootstrapper nembery$ python -m pytest␣
↪tests -v
============================================= test session starts␣
↪=============================================
platform darwin -- Python 3.6.5, pytest-3.5.1, py-1.5.3, pluggy-0.6.0 -- /Users/
↪nembery/PycharmProjects/panos_license_tool/panos-bootstrapper/bin/python
cachedir: .pytest_cache
rootdir: /Users/nembery/PycharmProjects/panos-bootstrapper, inifile:
collected 8 items

tests/test_bootstrapper.py::test_index PASSED                                  ␣
↪                    [ 12%]
tests/test_bootstrapper.py::test_caching PASSED                                ␣
↪                    [ 25%]
tests/test_bootstrapper.py::test_build_openstack_archive PASSED                ␣
↪                    [ 37%]
tests/test_bootstrapper.py::test_get_bootstrap_variables PASSED                ␣
↪                    [ 50%]
tests/test_bootstrapper.py::test_import_template PASSED                        ␣
↪                    [ 62%]
tests/test_bootstrapper.py::test_get_template PASSED                           ␣
↪                    [ 75%]
tests/test_bootstrapper.py::test_list_templates PASSED                         ␣
↪                    [ 87%]
```

(continues on next page)

(continued from previous page)

```
tests/test_bootstrapper.py::test_delete_template PASSED                    ␣
↪                    [100%]

============================================ 8 passed in 0.41 seconds␣
↪============================================
(panos-bootstrapper) DFWMACK0AJHTDG:panos-bootstrapper nembery$
```

CHAPTER

ELEVEN

ABOUT

The PAN-OS bootstrapper Utility is a tool to simplify the process of building bootstrap packages for Palo Alto Networks Next-Gen Firewalls.

Complete documentation on the Palo Alto Networks NGFW bootstrapping process can be found here.

An example web application is hosted on GitHub as the panos-bootstrapper-ui.

CHAPTER

TWELVE

ARCHITECTURE

This utility is provided as a micro-service that provides a simple API. It is expected that another application will consume this API for presentation to the user.

CHAPTER

THIRTEEN

CONTRIBUTING

Feel free to contribute templates, bug fixes, examples, documentation updates, etc to the GitHub repository. For simple contributions, opening an issue on GitHub is the preferred method. For more complex additions such as bugfixes, fork the project, commit your changes and open a Pull Request. Questions can be posed to the #automation-bof channel in the PaloAltoNetworks slack channel.

CHAPTER

FOURTEEN

INDICES AND TABLES

- genindex
- modindex
- search